Collusion-Resistant Watermarking (CREW)

Demba Ba

Hektor Malo

Nathanael Briggs

1. Introduction

Digital fingerprinting is a technique for identifying users who use multimedia content for unintended purposes, such as redistribution. Fingerprints can be embedded into digital content using watermarking techniques that are designed to be transparent and robust to a variety of attacks such as compression, noise addition etc…

However, an easy, yet cost-effective attack against watermarking is collusion. Collusion is defined as the fact of combining several marked copies of the same multimedia content, in an attempt to 'destroy' the underlying fingerprints. What this means is that if a known set of watermarks were embedded into multimedia content, a copy generated through collusion would be unidentifiable in terms of the watermark it contains. A simple but very practical example of where collusion could be used as described above is in the video entertainment industry. Say eight watermarked copies of an unreleased video were distributed to eight different professionals for technical reviewing. Collusion would be an easy way for the reviewers to leak out the video before its release date, without being caught. One therefore understands the need for techniques that would allow catching the people involved in generating a new copy of watermarked digital content through collusion.

Responding to that concern, Min Wu and K. J. Ray Liu of the University of Maryland (College Park) introduced Anti Collusion Codes (ACC's) as an effective way of fighting collusion. ACC's have the property that the composition of any subset of *K* or fewer code-vectors is unique. Using this property, it is possible to identify groups of *K* or fewer colluders. In our case, we are interested in *AND*-ACC's, where a group of colluders generates a new copy by taking the average of their respective copies. It can be shown that the averaging of their respective copies corresponds to the *AND* logical operation. To identify the group of colluders, we will use a detection algorithm that is based upon hard thresholding.

The remainder of the paper will touch the mathematical background and the basic implementation of ACC's in gray-scale and color images as well as audio through the conventional programming language, MATLAB. The development details of the ACC Embedder, Colluder and Detector will be discussed with an overview of the complex GUI interface that was developed to integrate all of the above components in a friendly manner.

2. ACC's in a nutshell

ACC's are the building blocks of the algorithms we developed, which allow us to catch a given number of potential colluders. In this section of the paper, we provide the reader with the necessary tools to understand the way our algorithms function. A rigorous mathematical proof on the basis of ACC's and how they are used to counter collusion attacks is provided in [1].

A $(v, k, 1)$ ACC uses an orthogonal basis U of size v to catch a maximum of $k - 1$ colluders. The maximum number of users that can be accommodated by the basis U and still catch $k - 1$ colluders is $n = (v^2 - v)/(k^2 - k)$. Such a code is called a $(k - 1)$-resilient ACC. It is also characterized by a v-by-n matrix C of 1's and 0's called the code matrix. Through the linear transformation given by $f : x \mapsto 2x - 1$, we obtain a matrix B = f(C) whose columns correspond to users 1, 2…, n. Indeed, the jth column of the B matrix gives the coordinates of the watermarking sequence w_j for user j in the orthogonal basis U.

We illustrate all of the above through an example. The following is the code matrix C for the (7, 3, 1) ACC. It can accommodate 7 users and catch two colluders:

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Each column of C is mapped to +/-1 through the *f* transformation, which results in the B matrix given as follows:

$$B = 2*C - 1 = \begin{pmatrix} -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{pmatrix}$$

The matrix U of orthogonal vectors is specified below. We chose not to specify the number of columns of U; it will become clear why when we discuss the implementation of ACC codes for different types of media (image, audio):

$$U = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{pmatrix}$$

As previously mentioned, each column of B specifies the watermarking sequence for each of the seven. The watermarking sequence w_j for user j is given as follows:

$$\mathbf{w}_j = \sum_{i=1}^{7} b_{ij}\mathbf{u}_i$$

For instance, the watermarking sequences for users 1 and 2 are:

User1: $\mathbf{w}_1 = -\mathbf{u}_1 - \mathbf{u}_2 + \mathbf{u}_3 - \mathbf{u}_4 + \mathbf{u}_5 + \mathbf{u}_6 + \mathbf{u}_7$

User2: $\mathbf{w}_2 = -\mathbf{u}_1 + \mathbf{u}_2 - \mathbf{u}_3 + \mathbf{u}_4 + \mathbf{u}_5 - \mathbf{u}_6 + \mathbf{u}_7$

The watermarks for the remaining five users can be written and embedded in the same manner so that the generated copies of the specific multimedia content are resistant to collusion involving two users.

The above example can be extended to more users. Although the number of users that can be accommodated by a particular ACC can be very large, it is not arbitrary. Indeed, it depends on the parameters of the code. For instance, if we want to catch 3 colluders (*k = 4*), the following relationship has to be satisfied:

$$v \equiv 4 \bmod 12 .$$

$v = 16$ is a solution to the above congruence, which means that a maximum of 20 users can be provided with an ACC code with such parameters.

In the remaining part of the paper, we discuss the implementation of our Collusion-Resistant Watermarking scheme.

3. Collusion by Averaging

The rapid development of computer networks and the increased use of multimedia content via the internet have resulted in faster and more convenient exchange of digital information. With the ease of editing and perfect reproduction, protection of reproduction, protection of ownership and prevention of unauthorized manipulation of digital images became important concerns. To respond to such concerns, multimedia watermarking schemes were introduced as a way to uniquely identify multimedia content. However, most invisible watermarking schemes are prone to collusion attacks under a very general framework. As described in the Embedder part, ACC's are a technique used to identify ownership and protect owner's rights in digital distribution.

Collusion is defined as the fact of combining several watermarked copies of the same multimedia content in an attempt to 'destroy' the underlying fingerprints. In a collusion attack, a group of colluders collectively obtain an average of their individually watermarked copies and try to escape from being identified. Due to the large number of variations of collusion attacks, it is hard to prove that the ACC-based watermarking scheme is resistant to all sorts of collusion attacks. However, the ACC watermarking scheme survives benchmark attacks when collusion by averaging is used. The number of colluders that can be caught depends on the parameters of the particular ACC's. In our MATLAB program presented in APPENDIX A, three traitors collude to generate a pirated image from watermarked images W_1, W_2 and W_3 generated from the same original image in the following way.

- We take three pixels z_1, z_2,z_3 from the same location of the images W_1, W_2 and W_3, respectively.
- We construct a pixel value $z = £(z_1, z_2, z_3)$ where £ is taken to be the average function. Therefore, in our case, $z = (z_1 + z_2 + z_3)/3$. In the same line, for future implementations, £ can be used to represent median, max value, minimum value or weighted average.
- We construct the image ***I*** with all the pixel values z.

A schematic diagram is shown in the figure below.

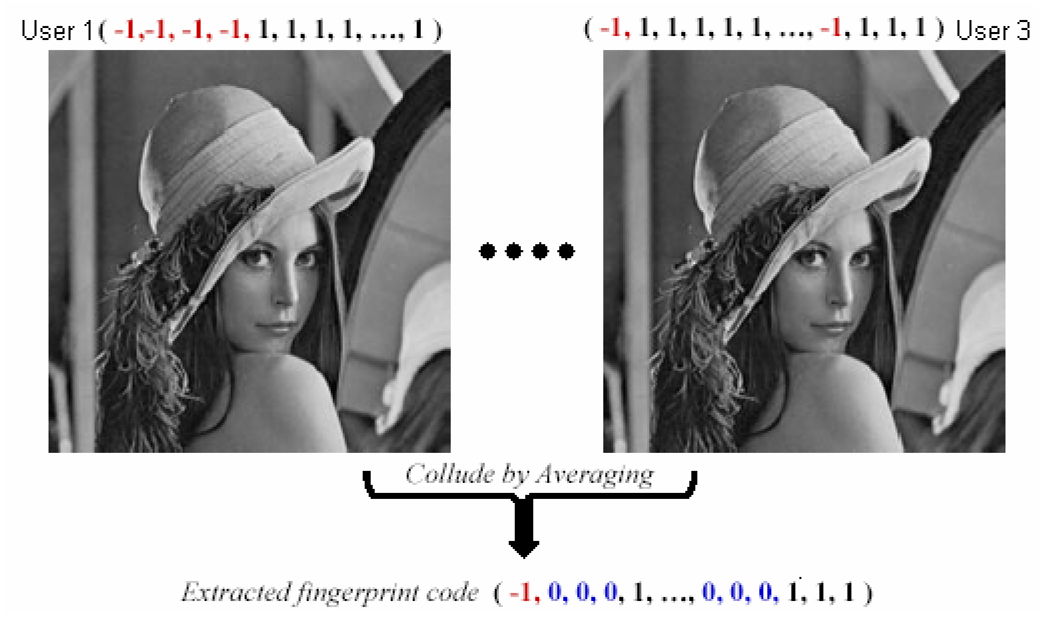

User 1(-1,-1, -1, -1, 1, 1, 1, 1, ..., 1)
(-1, 1, 1, 1, 1, 1, ..., -1, 1, 1, 1) User 3
Collude by Averaging
Extracted fingerprint code (-1, 0, 0, 0, 1, ..., 0, 0, 0, 1, 1, 1)

Figure 1. Illustration of Collusion by Averaging

Using this attack, the correlation factor between the traitors' key and the detected key is much greater than 50% while the correlation between the key of an innocent user and the detected key is very close to 50%. Therefore, the traitors can be identified and no one will be wrongly implicated. The tests that we conducted showed that our ACC watermarking scheme was resistant to collusion by averaging.

4. CREW – Collusion Resistant Watermarking

CREW concentrates on resilience to an attack that is especially applicable to the watermarks embedded into different multimedia. Collusion occurs when collections of images from different users are analyzed or combined with the ultimate goal of producing a mark-free copy of the original. Given watermarked image which is suspected under attack, the detection process traces back the original image by using correlation measure. This method, as will be illustrated in the rest of our paper, can survive the non-linear geometric attacks, common image transformations and intentional attacks both in spatial and frequency domain. The presentation of CREW scheme can stand certain collusion attacks such as average, minimum and maximum attacks.

To date, the collusion attacks have not been well studied, most probably because of the research focuses on still audio watermarking, where it does not arise in this form. However, its growing importance is evidenced by the publication of recent papers concentrating on collusion attacks. In the rest of this research paper, we specify the mathematical background of the embedding and detecting process without skipping the multiple-image collusion problem.

4.1 Embedder

Advances in multimedia sharing techniques were accompanied by one major concern: How can the intellectual property of digital content be protected? Digital watermarking techniques stood very early as an efficient and cost-effective way of addressing the issue. Digital watermarking describes a class of techniques that are used to embed copyright and other protective information in multimedia. The block diagram below shows how digital watermarks can be embedded in images.

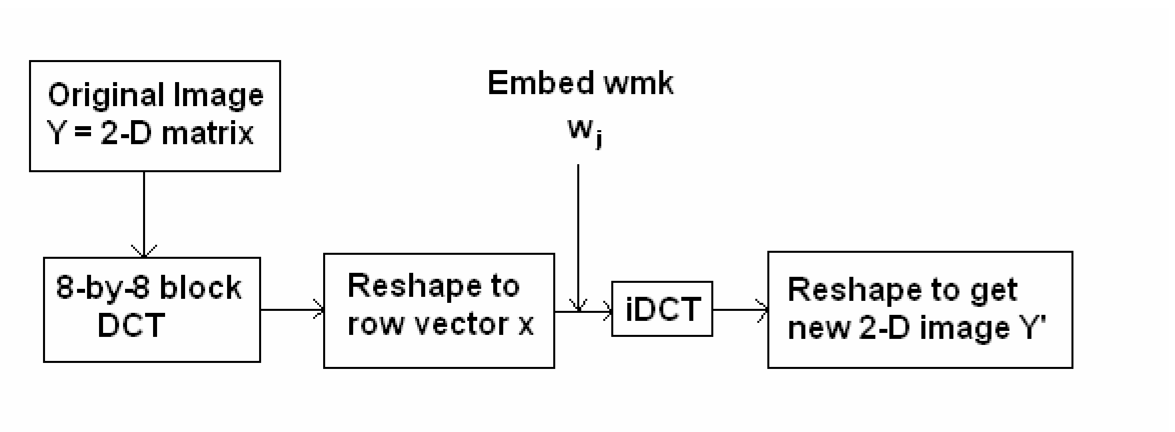


Figure 2. Block Diagram of the CREW Embedder

The embedding process is pretty self-explanatory from the above block diagram. However, there are a few implementation issues that should be addressed. As mentioned before, our collusion-resistant watermarking scheme was tested on images. This dictates both the size of the orthogonal basis U and limits the places in which we can embed.

After taking the 8-by-8 block DCT of the original image Y, we reshape it and call it x. If size(Y) = [M N], size(x) = [1 M*N]. This tells us that the number of columns of U should be M*N. Moreover, it is a known fact that most of the information contained in an image is stored in the DC values of x. That's why, one should be careful not to modify those values as it would lead to changes in the image that would be noticed by the human eye. To avoid the latter, we mask the positions of U that corresponds to the DC values of x and then embed. Then, we perform an inverse-DCT and reshape the row vector to obtain a 2-D matrix, which is our new image. As expected, the new image cannot be distinguished from the original image by simply looking at it.

4.2 Detector

The implementation of the ACC detector is the most important part of our project. We introduce an efficient detection algorithm for identifying the fingerprints associated with ***K*** colluders that require ***O(K*log(n/K))*** correlations for a group of ***n*** users. There are several detection schemes that we could have used but the fastest and the easiest one is a simple detection scheme based upon hard thresholding. The goal of this scheme is to find efficient detection structures by taking the advantages of the special characteristics of the ACC. By giving a brief mathematical background of the detection process, the reader will digest the ideas behind our detection code.

First, assuming that the total distortion **d** is an N-dimensional vector following an identically independent distributed Gaussian distribution with zeromean and variance σ_d^2. When ***K*** colluders come together and perform a collusion attack by averaging, as discussed in the previous section, they produce a colluded version of the original content. The colluded file denoted by **y** will become

$$\mathbf{y} \;=\; \frac{1}{K}\sum_{j\in S_c}\mathbf{y}_j + \mathbf{z} = \frac{1}{K}\sum_{j\in S_c}\mathbf{s}_j + \mathbf{d}$$

where **K** is the number of colluders, $\mathbf{y_j}$ the watermarked media, **d = x + z** represents the distortion signal, and **Sc** indicates a subset with size **K**. The market content $\mathbf{y_j}$ for each user **j** is given in the following equation,

$$\mathbf{y}_j = \mathbf{x} + \mathbf{s}_j = \mathbf{x} + \alpha\sum_{i=1}^{v} b_{ij}\mathbf{u}_i$$

where α controls the strength of the fingerprint.

The orthogonal basis $\mathbf{u_i}$ is used to calculate the correlator vector $\mathbf{T_N}$, with ith component given by the expression

$$T_N(i) = \mathbf{y}^T\mathbf{u}_i \Big/ \sqrt{\sigma_d^2 \cdot \|\,\mathbf{u}_i\,\|^2}$$

for **i** = **1, ..., v**. Another way to show the $\mathbf{T_N}$ vector is to indicate the colluders via the location of components whose values are 1 by **Φ** vector as shown in the equation below

$$\mathrm{T}_N = \frac{\alpha_1}{K}\mathbf{B}\mathbf{\Phi} + \mathbf{n} \qquad \alpha_1 = \alpha\sqrt{\|\,\mathbf{u}\,\|^2 / \sigma_d^2}$$

where **B** is the derived code matrix and K the number of 1’s in **Φ.**

Applying hard thresholding to the detection statistics $\mathbf{T_N(i)}$, we obtain a vector **Γ** = (Γ_1, Γ_2, …, Γ_v), where Γ_i = 1 if $\mathbf{T_N(i)}$ > **ς** (threshold value) and Γ_i = 0 otherwise. We determined the vector $\mathbf{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_n)^T \in \{0,1\}^n$ that describes the colluders set via the location of the components of $\mathbf{\Gamma}$ whose value are 1, so if $\Phi_j = 1$, then the jth user is the suspected colluder.

The Matlab algorithm that we designed, we used the fact that the element-wise multiplication "." of the binary vectors corresponds to the logical AND operation. The algorithm decides and updates the number of 1 in the $\mathbf{\Gamma}$ and uses AND operation to find the particular colluders. One of the main characteristics of the embedder is that it is very reliable even in the presence of noise. Suppose that the receiving colluded picture we obtain comes with White Gaussian Distributed Noise. We performed tests with noise where we were able to get nearly perfect detection of colluders.

4.3 Graphical User Interface (GUI)

4.3.1 Collusion-Resistant Watermarking (CREW) GUI

A GUI was designed for implementing CREW's in digital images using the MATLAB GUIDE toolbox. The motivation for this portion of the project is to provide a tool that any user, regardless of technical expertise, can easily use to secure their digital media. In addition, the CREW GUI is intended to be available for widespread distribution, and, therefore, the visual appeal of the GUI layout was also emphasized.

The design of the CREW GUI required several factors to consider. Table 1 below offers a list of original constraints and future possibilities for the design of the current CREW GUI.

Original Constraints	**Future Possibilities**
Modular relationship between components	GUI windows for collusion/detection
Implementation for .bmp files	Availability for more than 20 users
Availability for at least 7 users	Extended detection report
Source image display	
Saving embedded and colluded files	
Detection report	
Attractive color scheme	
Attractive appearance	

Table 1. Original Constraints and Future Possibilities of the CREW GUI

4.3.2 Description

The CREW GUI primarily consists of the following functional components: a help dialog, mode selection, a load function, a display function, a list of possible users, an embedder, a colluder, and a detector. An explanation of each functional component is given below. The flowchart in Figure 4 describes the ideal order in which to operate the CREW GUI.

The first element of the CREW GUI is the help dialog box, which is available for simple informational purposes. The content of this dialog box is not expansive in any sense, but it provides the user with enough guidance to navigate through the CREW process easily. In future iterations of this program, the help function will provide much more information for the user in terms of component functionality and troubleshooting. Mode selection is the decision whether to run the CREW in an audio or image file.

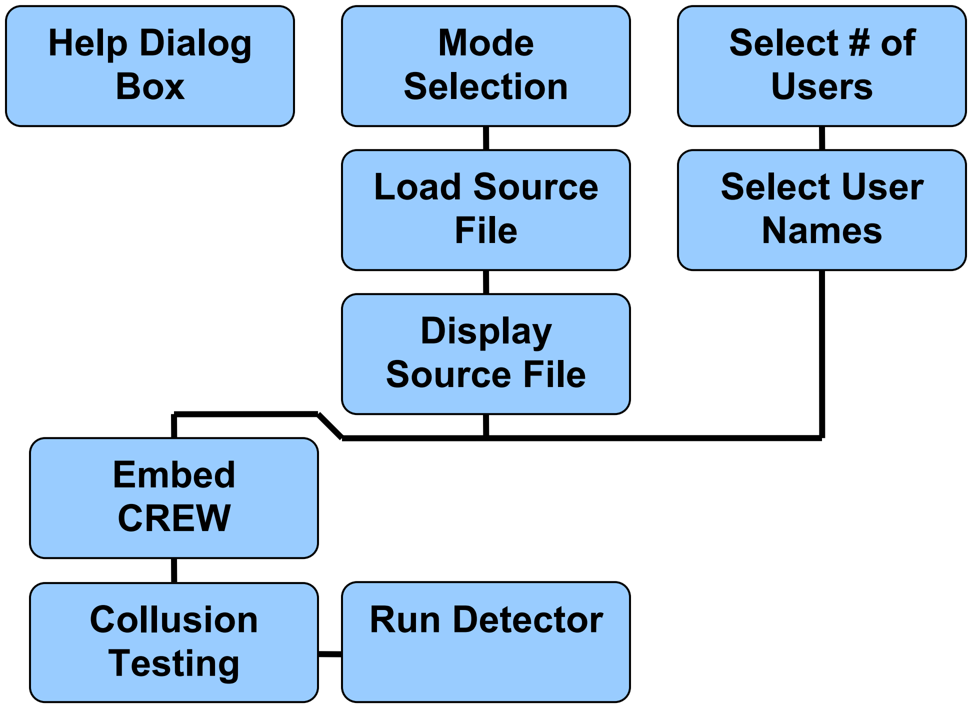


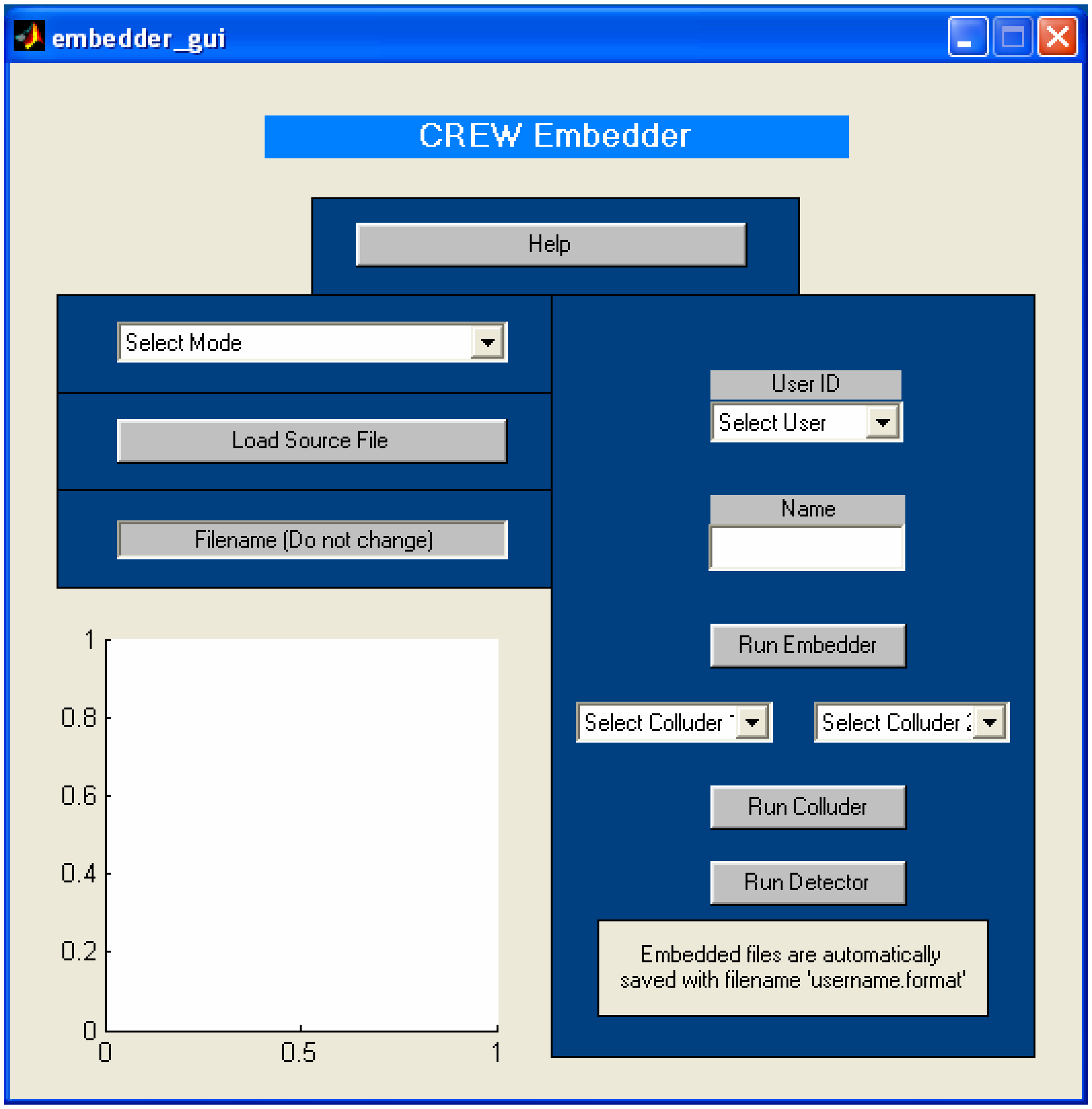
Figure 3. Flowchart Displaying the Order of Operations of the CREW GUI

In order to initialize the CREW process, an image must be selected and loaded into the GUI. Using the plot function, MATLAB is able to read the image and display it in the set of axes located in lower left corner of the interface. The filename is shown above the image to specify its identity. In the current version, only .bmp files (Windows bitmap format) are compatible with the embedder, colluder, and detector. A Windows load dialog box appears when the “Load Source File” button is pressed. Different source files can be loaded after one another, but only one source file at a time can be modified by the CREW scheme.

Once a source image has been loaded, the distribution list must be specified. A pull-down box on the right-hand side of the GUI offers a list of sixteen users from which to choose. Only eight users can be chosen for distribution of a color picture, and two colluders can be identified; sixteen users can be chosen for distribution of a black and white image, and three colluders can be identified. A unique name can be given to each user by typing the name in the editable text box below the pull-down menu. Names can be changed or deleted if necessary. However, in order for the CREW embedder to function correctly, the list of users must be continuous and in order. For example, if one wants to distribute an image to four users, the user list must contain *User01* through *User04* with appropriate names specified for each.

After the distribution list is complete and the source file has been loaded, the CREW is ready to be implemented. *Run Embedder* provides the users on the distribution list with uniquely watermarked images. The current version of the CREW GUI includes a collusion function (*Run Colluder*) that emulates a pair of users who collude two images and saves the resulting image in *colluded.bmp*. The *Run Detector* button runs the detection function which identifies the colluding pair. A message box which identifies the collusion suspects pops up upon completion of the detection phase.

A screenshot of the current CREW GUI is given in Figure 5. The layout was designed to best emulate the flowchart in Figure 4. It is fully functional for eight or fewer users for color images, and for sixteen or fewer users for black and white images.



Embedded files are automatically saved with filename 'username.format'

Figure 4. Screenshot of the Current CREW GUI

4.3.3 Future Modifications

The current scheme can be easily modified to perform a variety of new tasks. Some additional options include an increase in the number of users on the distribution list; a separate program for the colluder function; an extended help dialog; translation to another platform (i.e., Java, C/C++, etc.); addition of different variety of image files (e.g., .jpg, .gif, etc.); and further improvement on the visual appeal of the GUI.

5. Criterion Testing of the Engineering Design Process

For our project, we did an ethical analysis of its design criteria. We looked at several options for each different criterion. The analysis will be fully explained below.

Table 2 shows the various components of the design process that we had to analyze. In each row, a value greater than unity means that the component in the column is considered more vital for design than the component in the row. For instance, in the cell (1,3), *computing power* is considered to be less desirable to be improved upon than *add-on* features.

Criterion	c_1: computing power	c_2: optimization	c_3: add-on features	C_4: size on disk	c_5: proprietary code	c_6: code security
c_1: computing power	1.00	1.00	0.33	1.00	1.00	0.25
c_2: optimization	1.00	1.00	5.00	1.00	0.13	0.33
c_3: add-on features	3.00	0.20	1.00	0.50	0.20	1.00
c_4: size on disk	1.00	1.00	2.00	1.00	1.00	0.13
c_5: proprietary code	1.00	8.00	5.00	1.00	1.00	5.00
c_6: code security	4.00	3.00	1.00	8.00	0.20	1.00
Totals	11.00	14.20	14.33	12.50	3.53	7.71

Table 2. Criterion Comparison Table.

An explanation for each criterion is now given. *Computing power* refers to the ability of the code to perform complex operations on files. *Optimization* refers to the speed of running the algorithm on an average computer. *Add-on features* are additional modules that provide other services and options, such as separate windows for the colluder and detector. *Size on Disk* refers to the size of the program on a hard drive. *Proprietary Code* is a term that refers to code that is originally written by our team rather than using prefabricated MATLAB functions. *Code Security* is the level of safety that allows only the users of our program to know how to decode the watermarks.

Each of these components have been weighted against each other in Table 2. Table 3 is a chart displaying the computation of the criterion weights. In addition, the best design will be found according the rank of each criterion it receives from this calculation.

	c_1	c_2	c_3	c_4	C_5	c_6	Row Total	Nrm Wts.	Rank
c_1: computing power	0.09	0.07	0.02	0.08	0.28	0.03	0.58	0.096785	2nd
c_2: optimization	0.09	0.07	0.35	0.08	0.04	0.04	0.67	0.111479	4th
c_3: add-on features	0.27	0.01	0.07	0.04	0.06	0.13	0.58	0.097174	6th
c_4: size on disk	0.09	0.07	0.14	0.08	0.28	0.02	0.68	0.113462	5th
c_5: proprietary code	0.09	0.56	0.35	0.08	0.28	0.65	2.02	0.335911	1st
c_6: code security	0.36	0.21	0.07	0.64	0.06	0.13	1.47	0.24519	3rd
Totals	1.00	1.00	1.00	1.00	1.00	1.00	6.00	1	

Table 3. Computing Criterion Weights.

With the computations above, we can find the best design using our criteria. Table 4 displays this calculation. The result seems to be that design #4 seems to be the best with the given weights. It leans heavily on *Computing Power*, *Add-on Features*, and *Code Security*. Thus, our program should be able to handle a decent amount of computations while maintaining software security. In addition, a few more features should be made available to the user. Our current design handles most of these criteria well, but there is a lack of add-on features, which require much more time to develop than just the code itself.

		Designs			
	Weight	d_1	D_2	D_3	d_4
Aggregated Score		0.217179512	0.2651744	0.3396302	0.408105369
c_1: computing power	0.223647946	0.2	0.4	0.6	0.8
c_2: optimization	0.125562792	0.8	0.6	0.4	0.2
c_3: add-on features	0.056084582	0.2	0.4	0.6	0.6
c_4: size on disk	0.106123088	0.1	0.1	0.2	0.2
c_5: proprietary code	0.316972362	0.05	0.05	0.1	0.2
c_6: code security	0.171609229	0.2	0.3	0.4	0.5

Table 4. Results from the Criteria Testing.

Our program most likely emulates design #2. However, the integrity of each of the components is not critical for our application. The implementation of the CREW Embedder is much more vital just for its functionality than its marketability at this point. In the future, if a CREW Embedder became a marketable commodity, we would need to implement design #4 for best results.

This treatment of criterion testing fulfills element #2 of the IEEE Code of Ethics because we desire to have full security in our watermarking procedure. The person who buys the code should have full protection of their work, and conflicts of interest should not inhibit our distribution of this watermarking technology. Thus, everyone will have a fair chance of implementing CREW's and detecting violators.

6. Conclusion

In this paper, we have discussed and proved the effectiveness of ACC's in countering collusion attacks. ACC's have the property that the composition of any subset of *k* or fewer code-vectors is unique, which allows us to catch groups of *k* or fewer colluders. The codes, which were tested on gray scale and color images, were found to be successful in catching the desired number of colluders. A Graphical User Interface (GUI) was also designed to allow integration of the embedder and the detector in a single user-friendly environment.

Currently, we are investigating the use of ACC's in audio. One of the challenges we have faced so far is the proper embedding of watermarks to avoid noise in the generated copies. Also, it appears that hard detection is not appropriate for this type of media, which led us to turn our attention to another detection scheme called the Adaptive Sorting Approach (ASA). One drawback of hard detection is that the threshold is set at the beginning of the algorithm, regardless of statistical observations. The ASA could provide a solution to that problem.

REFERENCES

1. W. Trappe, **M. Wu**, Zhen Wang, K.J.R. Liu: "Anti-collusion Fingerprinting for Multimedia", IEEE Trans. on Signal Processing, Special issue on Signal Processing for Data Hiding in Digital Media & Secure Content Delivery, vol. 51, no. 4, pp.1069-1087, April 2003 .
2. **M. Wu** and B. Liu: "Data Hiding in Image and Video: Part-I -- Fundamental Issues and Solutions", IEEE Trans. on Image Proc., vol.12, no.6, pp.685-695, June 2003.
3. **M. Wu**: "Joint Security and Robustness Enhancement for Quantization Based Embedding," accepted by IEEE Trans. on Circuits and Systems for Video Technology, Special Issue on Authentication, Copyright Protection, and Information Hiding, May 2003.
4. **M. Wu** and B. Liu: *Multimedia Data Hiding*, Springer Verlag, ISBN#0387954260, October 2002.
5. L. Boney, A. H. TewfiE, and K. N. Hamdy. Digital watermark for audio signals. International Conference on Multimedia Computing and Systems, pp. 473-480, 1996.
6. J. Laeh, W. H. Mangione-Smith, and M. Potkonjak. FPGA Fingerprinting Techniques for Protecting Intellectual Property. Proceedings of CICC, 1998.
7. M. M. Yeung, E. C. Mintzer, G. W. Braudaway, and A. R. Rao. Digital watermarking for high-quality imaging. Workshop on Multimedia Signal Processing, pp. 357-362, 1997.